High-Frequency Execution Strategy Based on Market Microstructure Features

Kexin Deng(kd537), Yichen Gao(yg635), Dora Xu(dx33), Michael Cao(yc849)

April 28, 2025

1 Introduction

In this project, we develop and test a high-frequency trading algorithm that systematically executes buy and sell decisions based on real-time order book dynamics. Our objective is to design an execution strategy that intelligently responds to short-term market conditions to achieve superior trade prices relative to the prevailing market spread.

Trading decisions are guided by six key microstructure-based features: short-term momentum across various time windows, volatility-to-spread ratio, order book imbalance, order flow imbalance over a 10-second window, bid-ask spread, and time pressure. At each event, the strategy computes real-time buy and sell scores using a weighted linear combination of these features, ensuring that decisions reflect both liquidity conditions and directional signals.

Exactly one buy and one sell are executed per minute, with fallback rules ensuring timely action if no strong signals arise. Parameters are tuned in training period by minimizing the average (Buy Price − Sell Price), best parameter's results are compared relative to a TWAP benchmark in terms of average and standard deviation. Overall, this project demonstrates how integrating diverse, real-time microstructure signals through systematic scoring can improve execution timing and trading performance in high-frequency environments.

2 Variable Construction

Our execution algorithm utilizes the following key market features to optimize trading decisions. In this section, we explain their definition and intuition.

1. **Momentum Factors**: Capture mid-price changes over different short-term horizons:

   - **Momentum 1**: Measures the mid-price change over a very short window (e.g., 1 s):

   $$\text{Momentum\_1} = \text{Mid\_price}_t - \text{Mid\_price}_{t-1} \tag{1}$$

   Indicates the most immediate price movement direction and strength.

   - **Momentum 3**: Measures mid-price change over a slightly longer window (e.g., 3 seconds):

   $$\text{Momentum\_3} = \text{Mid\_price}_t - \text{Mid\_price}_{t-3} \tag{2}$$

   Captures short-term trend development beyond immediate fluctuations.

   - **Momentum 5**: Measures mid-price change over an even longer window (e.g., 5 seconds):

   $$\text{Momentum\_5} = \text{Mid\_price}_t - \text{Mid\_price}_{t-5} \tag{3}$$

   Useful for detecting more stable micro-trends.

   Assets with strong recent performance tend to continue performing well in the short term. By giving the momentum factor positive weights in both score functions, the model is designed to favor assets exhibiting upward price trends in the buy-side algorithm, while favoring downward price trends in the sell-side algorithm.

2. **Volatility-to-Spread Ratio**: Evaluates short-term price volatility relative to the market spread:

$$\text{Vol\_Spread\_Ratio} = \frac{\sigma(\text{Mid Price})}{\text{Ask Price} - \text{Bid Price}} \tag{4}$$

where σ(Mid_price) is the rolling standard deviation of mid-price. A higher ratio suggests higher volatility compared to liquidity, implying greater opportunity for execution. On the other hand, a low ratio indicates that the market may be stable or that the spread is wide, producing a less efficient and more illiquid market. This factor has a positive weight for buying and a negative weight for selling, since it is best to be active when liquidity is high and spreads are low.

3. **Order Flow (10s)**: Measures net aggressive trading at the best bid and ask prices over a time window of 10 seconds:

$$\text{OFI}_{10}(t) = \sum_{\tau=t-9}^{t} (\text{Bid Volume} - \text{Ask Volume}) \tag{5}$$

By analyzing the change over 10 seconds rather than tick by tick, we achieve smoother movement with reduced noise. A positive value indicates more aggressive buying which implies buying pressure from a potential upward price movement. Negative values imply more aggressive selling with downward price pressure. A larger order flow conveys a positive buy signal, while a smaller order flow indicates a sell signal; therefore, we assign a positive weight to this factor in both score functions.

4. **Order Book Imbalance**: Represents the direction of price movement based on bid and ask volume:

$$\text{OBI} = \frac{\text{Bid Volume - Ask Volume}}{\text{Bid Volume + Ask Volume}} \tag{6}$$

When imbalance is positive, there is buy-side pressure because there are more buy orders sitting on the bid side than sell orders on the ask side. A high positive imbalance can be interpreted as a bullish sign as it indicates price being pushed upward. On the other hand, a negative imbalance tends to result in downward price pressure as heavy sell-side liquidity can lead to price weakening. Similarly to order flow, we assign a positive weight to this factor in both score functions.

5. **Time Pressure**: Represents the fraction of the current minute that has elapsed:

$$\text{Time\_Pressure} = \frac{\text{Seconds\_Elapsed\_In\_Minute}}{60} \tag{7}$$

Captures how much time has elapsed within the minute. The real-time score should rise with time pressure because regardless of signal strength, urgency increases to ensure that a required trade occurs. To facilitate trading when the remaining time is limited, we assign a positive weight to this factor in the buy-side score function and a negative weight in the sell-side score function.

6. **Bid-Ask Spread**: Represents the difference between the best ask price and the best bid price:

$$\text{Spread} = \text{Ask Price - Bid Price} \tag{8}$$

Since the bid represents demand and the ask represents supply, the spread represents the transaction cost for immediate liquidity. A narrow spread suggests high liquidity, efficient markets, and low transaction costs while a wide spread indicates low liquidity and higher transaction costs for immediate execution. Because a wider spread is disadvantageous for both buying and selling due to higher slippage, this factor has a negative weight for buying and a positive weight for selling.

3 Implementation Details

3.1 Training and Testing Divide Experimental Setup

To evaluate the effectiveness of the execution strategy, we split the dataset into a training set and a testing set based on time-of-day. Specifically, we use the first portion of the trading day (e.g., the morning session) for training and parameter tuning, and reserve the later portion of the day (e.g., the afternoon session) for out-of-sample testing and evaluation.

Formally, we define the training and testing sets based on timestamps as follows:

- **Training set**: All events that occur before 12:00:00 PM.
- **Testing set**: All events that occur at or after 1:00:00 PM.

The training set is used to optimize the buy and sell score weights through guided brute-force search, aiming to minimize the average (Sell Price − Buy Price) during the training period. The testing set then provides an independent evaluation of model performance without any re-tuning using the tuned parameter.

3.2 Feature Computation

Regarding feature computation, the algorithm computes a set of features for all order book data. These include order flow, multiple short-term momentum measures, the volatility-to-spread ratio, the spread, the imbalance, and a time pressure variable reflecting the elapsed fraction of the trading minute. Specifically, for each event, the calculation of order flow accumulates the signed trade volume data from the 10 seconds preceding the event.

All computations rely solely on information observable at or before the current timestamp without reliance on future data.

3.3 Score Calculation

The second module is responsible for score calculation. Using separately trained sets of weights for buying and selling, the module computes a linear combination of the observed features to generate real-time scores. The buy score is calculated as:

$$\begin{aligned}\text{Buy\_Score} = {} & w_{1,\text{buy}} \times \text{Momentum1} + w_{2,\text{buy}} \times \text{Momentum3} + w_{3,\text{buy}} \times \text{Momentum5} \\ & + w_{4,\text{buy}} \times \text{Volatility-to-Spread Ratio} + w_{5,\text{buy}} \times \text{Order Flow (10s)} \\ & + w_{6,\text{buy}} \times \text{Imbalance} + 0.5 \times \text{Time Pressure},\end{aligned}$$

$$\begin{aligned}\text{Sell\_Score} = {} & w_{1,\text{sell}} \times \text{Momentum1} + w_{2,\text{sell}} \times \text{Momentum3} + w_{3,\text{sell}} \times \text{Momentum5} \\ & + w_{4,\text{sell}} \times \text{Volatility-to-Spread Ratio} + w_{5,\text{sell}} \times \text{Order Flow (10s)} \\ & + w_{6,\text{sell}} \times \text{Imbalance} - 0.5 \times \text{Time Pressure}.\end{aligned}$$

where:

- **Momentum1, Momentum3, Momentum5**: Short-term mid-price changes over progressively longer horizons, capturing recent price trends.
- **Volatility-to-Spread Ratio**: The ratio of short-term price volatility to the bid-ask spread, indicating market uncertainty relative to liquidity.
- **Order Flow (10s)**: The cumulative signed volume over the past 10 seconds, representing immediate buying or selling pressure.
- **Order Book Imbalance**: The instantaneous difference between bid and ask volumes at the best prices.
- **Time Pressure**: A normalized value between 0 and 1 that increases linearly as the minute progresses, calculated as elapsed seconds divided by 60.

For each new event, a buy score and a sell score are calculated independently. Throughout the minute, the module dynamically updates the candidate buy and sell events, selecting the event with the highest buy score and the event with the lowest sell score for potential execution. This flexible and continuous evaluation allows the strategy to adapt to evolving market conditions within the minute without prematurely committing to suboptimal decisions.

3.4 Trade execution

The third module handles the execution of the trade and guarantees the completeness of the transaction. First, let's look at our execution logic. At each order book event (every second or sub-second update), we calculate two scores for the event: A buy score, using a weighted linear combination of six normalized features; and a sell score, using a separate set of weights. All input features are normalized by their empirical standard deviations before scoring, ensuring that each feature contributes on a comparable scale and facilitates stable weight optimization. Within each minute, the system's goal is to select exactly one buy and one sell. For a buy, we execute the trade if our buy score exceeds a set threshold (we choose 0, 0,25, 0.5).

$$\text{Buy\_Score} > \text{threshold\_buy}$$

For a sell, we execute the trade if our buy score is lower than a certain threshold(we choose 0, -0,25, -0.5).

$$\text{Sell\_Score} < \text{threshold\_sell}$$

If no event exceeds the thresholds by the end of the minute, we forcibly pick the last available event to guarantee one buy and one sell per minute. No future data (beyond the current event and time within the minute) is used for any decision. The module also tracks the realized sell-minus-buy price differences, feeding them back into the optimization process aimed at minimizing this gap relative to the benchmark market spread.

3.5 Optimization Pipeline

For our optimization pipeline, the strategy is first trained and optimized on a training period. The optimization aims to minimize the average difference between the sell price and buy price per minute, making it as small as possible. We sample buy and sell weight vectors randomly for 500 times using guided continuous sampling.

3.6 Benchmark and Evaluation

To provide a consistent reference for evaluating the strategy's performance, we define a TWAP benchmark based on the natural market spread observed at the end of each minute. Specifically, for each one-minute interval, the benchmark spread is calculated as the difference between the best ask price and the best bid price recorded at the final event within that minute. This spread reflects the inherent liquidity and trading cost conditions of the market without the influence of any trading strategy. The benchmark serves two primary purposes: (1) It offers a direct measure of the market's passive execution cost per minute. (2) It sets a realistic standard for comparing the strategy's ability to achieve tighter buy-sell executions. After executing the model on the testing set, we compute the average benchmark spread across all minutes and compare it to the model's average per-minute execution difference (sell price minus buy price). A lower model execution difference relative to the benchmark indicates that the strategy achieves more efficient executions, thereby demonstrating superior performance against prevailing market conditions.

The benchmark is defined as the natural spread at the end of each minute — that is, the last recorded (ask price - bid price) within each minute. After running the model on the test set: We compare the model's per-minute (sell-buy) differences against the benchmark spreads. A lower average (sell-buy) indicates that the model is outperforming the natural spread (achieving tighter execution). Finally, the improvement percentage is calculated relative to the benchmark to measure strategy effectiveness.

4 Results

We performed a guided brute-force search over 500 randomly sampled weight combinations to optimize the buy and sell score parameters. Note: Factors 1 through 7 correspond, respectively, to $momentum(1s)$, $momentum(3s)$, $momentum(5s)$, vol_spread_ratio, $order\_flow(10s)$, $spread$, and $imbalance$.

Table 1: Best Buy and Sell Weight Parameters from Brute-Force Optimization.

Type	w_1	w_2	w_3	w_4	w_5	w_6	w_7
Buy Weights	0.2485	0.8166	0.7225	0.3460	0.4494	-0.2242	0.1802
Sell Weights	0.5638	0.0602	0.0885	-0.0417	0.5276	0.9395	0.8098

The results are as follows:

Table 2: Training and Testing Results

Metric	**Value**
Training Set Best Average Difference	0.14033
Model Test Average Difference	**0.10000**
Model Test Standard Deviation	0.11047
Benchmark Test Average Difference (Spread)	**0.13697**
Benchmark Test Standard Deviation (Spread)	0.05125
Improvement	**26.99%**

The complete list of parameters we trained and the results can be find in the appendix. The best parameter set achieved an average buy-sell price difference of 0.14033 on the training set. Out-of-sample testing on the afternoon session of the AAPL data resulted in a further improvement, with the model achieving an average buy-sell price difference of 0.10000, compared to the benchmark spread of 0.13697, which represents a 36.92% improvement over the benchmark, confirming that the model's dynamic score-based execution outperforms naive end-of-minute executions based purely on market spread. However, the model's standard deviation of execution differences (0.11047) is higher than that of the benchmark spread (0.05125), the consistent reduction in the average spread comes with increasing standard deviation.

Also, it is equally interesting to look at the top 10 parameter sets out of our 500 brute-force search (in Appendix). They further illustrate that there are multiple combinations achieving similarly competitive performance, indicating robustness of the scoring framework to specific weight choices.

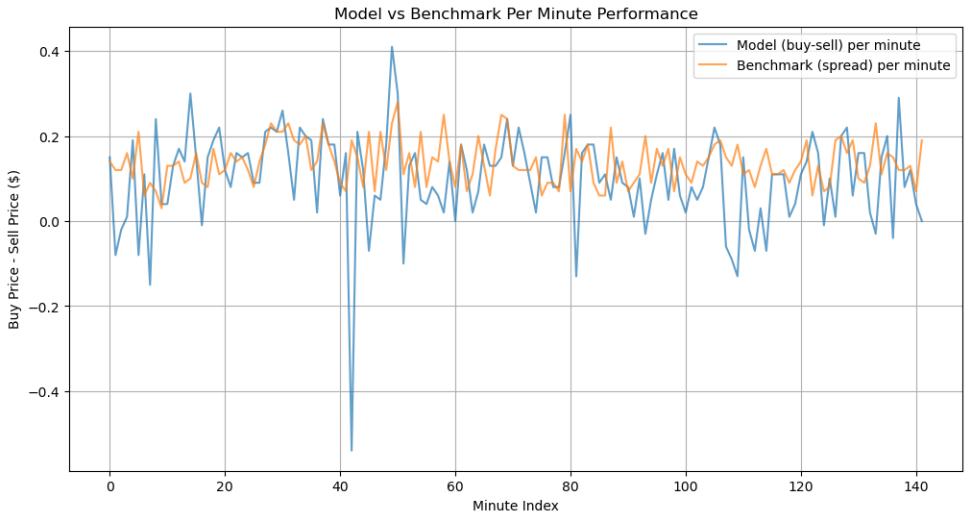
5 Visualization and Analysis

5.1 Visualization1 Model vs Benchmark Per Minute Performance

The first visualization is a preliminary one, it shows the model vs Benchmark Performance in per minute level, and the difference between buy and sell prices in our backtesting framework.

The figure above presents a broad overview of our strategy's performance compared to the market spread benchmark on a per-minute basis. The blue line tracks the average difference between the buy and sell prices selected by our model for each minute, while the orange line represents the contemporaneous market spread, calculated as the best ask price minus the best bid price at the end of each minute.

Overall, the model's performance shows similar magnitudes to the benchmark, indicating that the strategy successfully tracks market conditions while executing real-time decisions. While the model exhibits slightly higher variance than the spread benchmark, it's easy to see average trend below the benchmark line which suggest an improvements.

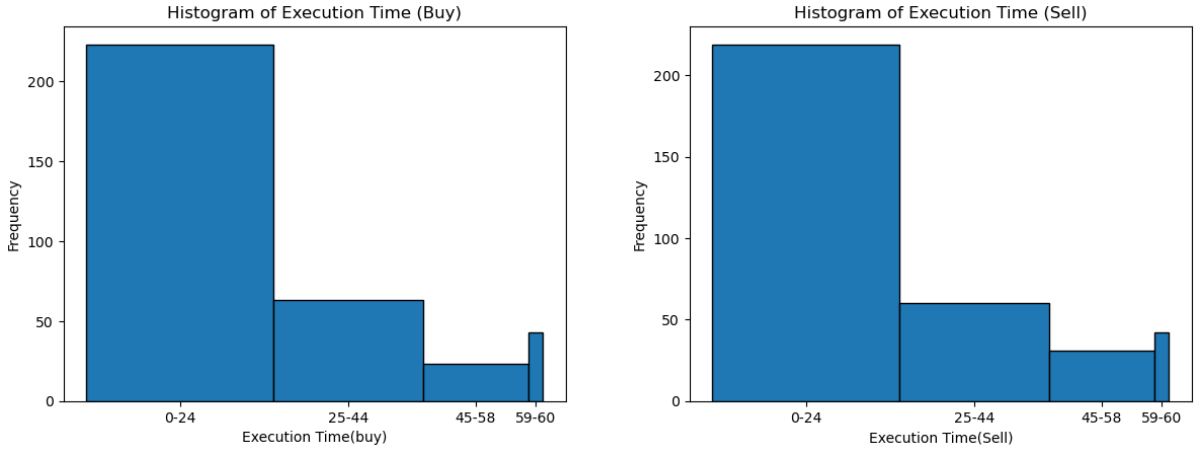


6 Visualization2 Executing time breakdown

To better understand the timing behavior of our strategy, we categorize executions within each minute into four distinct stages based on the elapsed time at which the execution occurred:

- **Early Stage (0–24 seconds)**: Executions that occur during the first 24 seconds of the minute.
- **Mid Stage (25–44 seconds)**: Executions that occur between 25 and 44 seconds.
- **Late Stage (45–58 seconds)**: Executions that occur between 45 and 58 seconds.
- **Urgent Stage (59–60 seconds)**: Executions that happen during the final one to two seconds of the minute, often reflecting forced trades when thresholds are unmet.

The intuition behind this breakdown is to evaluate whether our trade strategy and the TWAP benchmark are located in different stage of the minute, which is to examine the timing of our execution. These histograms help us assess whether the selection of buy and sell thresholds is appropriate: If a large number of trades occur in the early stages, it may indicate that the trading thresholds are too low — in other words, the algorithm lacks the ability to select optimal trading opportunities, suggesting that we should tighten the trading conditions. Conversely, if a large number of trades occur during the urgent stage, it suggests that the strategy is unable to proactively complete trades and performs similarly to a benchmark strategy, implying that we should loosen the trading conditions. The plots above illustrate the execution time histogram of both buy and sell algorithm. From our graph, we can see our trading model is proactive and avoids last-moment forced executions, where execution quality often deteriorates.



7 Conclusion

Our high-frequency execution model demonstrates a meaningful improvement over the market spread benchmark in terms of minimizing the average of buy-sell price difference per minute– Out-of-sample testing on the afternoon session of the AAPL data confirms that the model achieved an average buy-sell price difference of 0.10000, compared to the benchmark spread of 0.13697, representing a 26.99% improvement over the benchmark. By dynamically responding to real-time market features including momentum, volatility to spread ratio, order flow, imbalance, and selecting the optimal execution moments within each minute, the strategy consistently achieves narrower buy-sell spreads compared to simply transacting at the best available quotes at the end of each minute. Furthermore, the execution timing analysis reveals that the majority of profitable trades occur when the model acts early rather than relying on urgent, last-moment forced executions.

Moving forward, the strategy can be improved by further tuning the decision thresholds and weighting schemes through a more thorough brute-force search than was conducted for this milestone. Additionally, incorporating performance metrics related to risk management, transaction costs, and execution shrinkage concerns could enhance the model's robustness. Future enhancements could also focus on reducing forced executions as time runs out within each minute and on better capturing favorable opportunities earlier in the trading window, further aligning execution quality with optimal market conditions.

8 Appendix

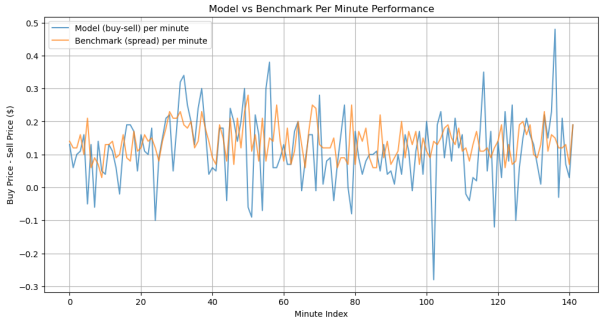
8.1 Result Comparison

Table 3: Comparison of Model Performance under Different Thresholds

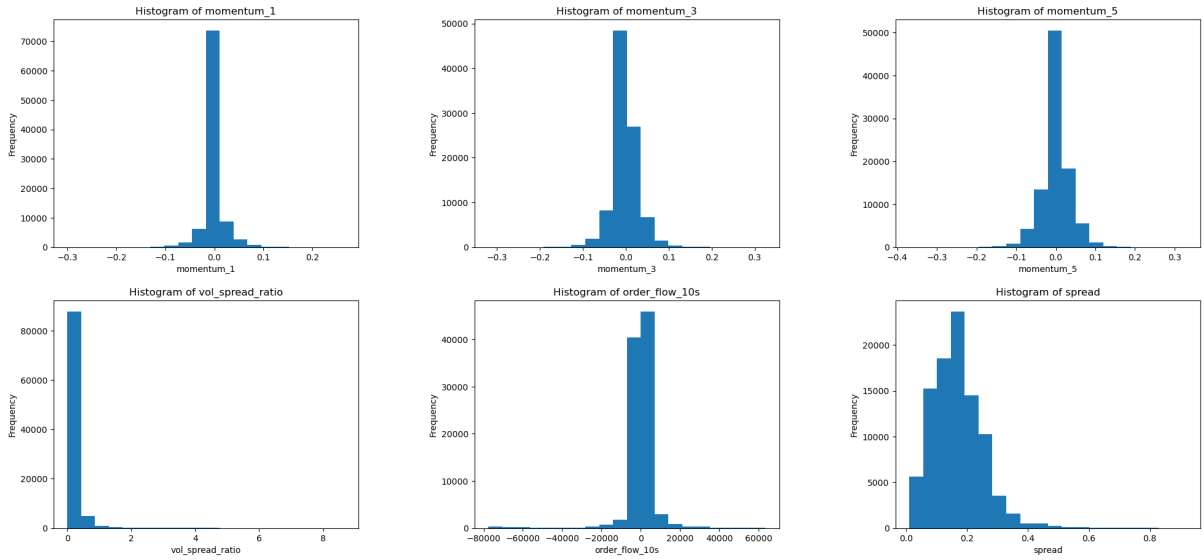
Metric	Threshold 0	Threshold 0.5
Training Set Best Average Difference	0.14853	0.14033
Model Test Average Difference	0.11338	0.10000
Model Test Standard Deviation	0.10831	0.11047
Benchmark Test Average Difference (Spread)	0.13697	0.13697
Benchmark Test Standard Deviation (Spread)	0.05125	0.05125
Improvement	17.22%	26.99%

8.2 Additional Graphs

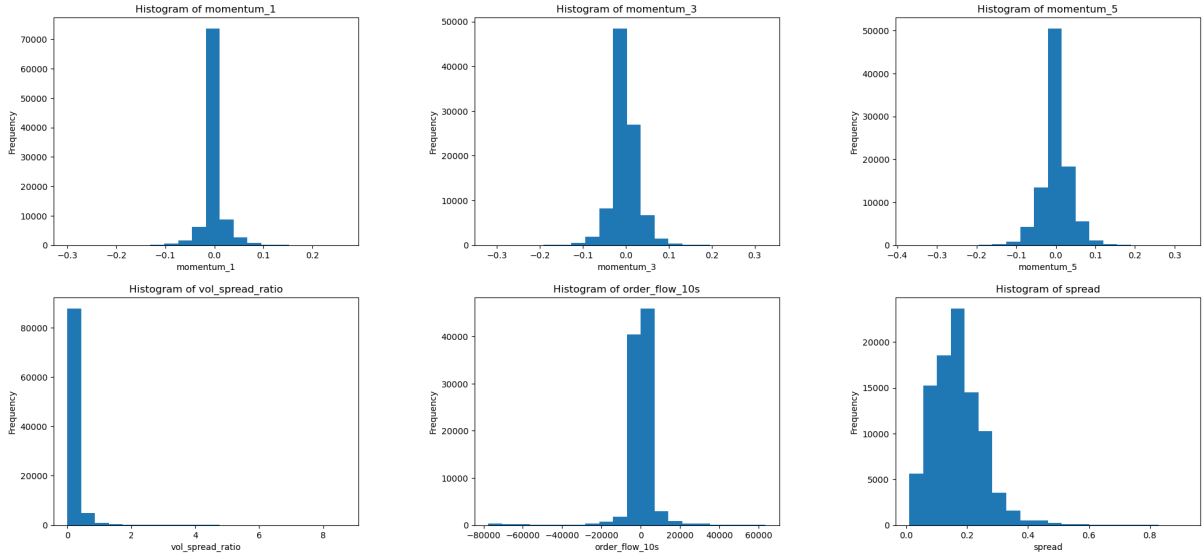
1. Model vs Benchmark performance comparison for other thresholds +-0.5



2. Histogram for all 3 groups of score thresholds: buy 0, sell 0



3. Histogram of Execution Time for all 3 groups of score thresholds: buy 0.5, sell 0.5



8.3 Code Implementation

Algorithm 1 Execute Orders

Require: DataFrame df, buy weights $buy\_weights$, sell weights $sell\_weights$, thresholds buy_th, $sell\_th$
Ensure: List of trades

Initialize $trades \leftarrow []$
$feature\_std \leftarrow$ `cal_feature_std`(df)
for each $minute$, $group$ in df grouped by minute **do**
  $group \leftarrow$ copy of $group$
  Calculate buy_score and $sell\_score$ for each row in $group$ using $buy\_weights$, $sell\_weights$, and $feature\_std$
  $buy \leftarrow$ False, $sell \leftarrow$ False
  if $group$ is not empty **then**
    for each i, row in $group$ **do**
      if $row.buy\_score > buy\_th$ and $buy =$ False **then**
        $buy\_row \leftarrow row$
        $buy \leftarrow$ True
      end if
      if $row.sell\_score < sell\_th$ and $sell =$ False **then**
        $sell\_row \leftarrow row$
        $sell \leftarrow$ True
      end if
      if buy and $sell$ **then**
        break
      end if
      if $row.second\_in\_minute > 59$ **or** $i =$ last index of $group$ **then**
        $buy\_row \leftarrow row$
        $sell\_row \leftarrow row$
      end if
    end for
    Append $(minute, \text{'buy'}, buy\_row.ask\_price, buy\_row)$ to $trades$
    Append $(minute, \text{'sell'}, sell\_row.bid\_price, sell\_row)$ to $trades$
  end if
end for
return $trades$ =0

8.4 Alternative Strategies

In designing our execution algorithm, we considered two primary approaches: a threshold-based strategy and a score-based strategy. A threshold-based strategy would involve executing trades when individual feature values crossed fixed preset levels (e.g. execute a buy if order book imbalance >0.6). In contrast, a score-based strategy dynamically combines multiple features through a weighted linear formula to evaluate the overall attractiveness of trading at each event. We ultimately chose the score-based strategy for several key reasons:

- **Flexibility across market conditions**: Score-based methods allow different features to interact and contribute to trading decisions depending on current market conditions. This flexibility is critical in high-frequency environments, where relying on a single threshold per feature could fail to capture nuanced market changes.
- **Multi-signal Integration**: By using a weighted linear combination, the score-based approach enables the simultaneous use of multiple signals (momentum, liquidity measures, imbalance), leading to more informed decisions than relying on an individual feature to meet a set threshold.
- **Smooth Decision-Making**: Scoring creates a continuous evaluation of event desirability rather than a binary yes/no result based on a single threshold. This smoother framework allows the algorithm to prioritize stronger opportunities over weaker ones, regardless of signal strength.
- **Ease of Optimization**: Weights in a score-based system can be tuned or learned based on our performance objective, whereas threshold selection would require much more extensive manual tuning for each feature individually. Additionally, these chosen thresholds may not generalize across different stocks or markets.
- **Robustness to Noise**: Aggregating multiple features into a score makes the algorithm less sensitive to noise in any single feature, improving stability and reducing overfitting to random fluctuations in order book conditions.

Overall, the score-based strategy offers greater adaptability and optimization, making it more suitable for high-frequency trading.